The Microsoft® Windows® Guidelines for Accessible Software Design

Designing and Building Applications That Are Usable by People with Disabilities

Please send comments or suggestions to:
Accessibility and Disabilities Group
One Microsoft Way
Redmond, WA 98052-6399

Contents i

Introduction to Accessibility

Personal computers are powerful tools that enable people to work, create, and communicate in ways that might otherwise be difficult or impossible. The vision of making computers easier for everyone to use, however, can only be realized if people with disabilities have equal access to personal computing.

Background About the Accessibility Issue

Computer accessibility is becoming an increasingly important issue in the home and workplace. An estimated eight out of ten major corporations employ people with disabilities who may need to use computers as part of their jobs. Over 30 million people in the U.S. alone have disabilities that can be affected by the accessibility of computer software. In addition, as the population ages, more people experience functional limitations, causing computer accessibility to become a more important issue for everyone.

Legislation in the U.S., such as the Americans with Disabilities Act (which affects private businesses with more than 15 employees) and Section 508 of the Rehabilitation Act (which affects the federal government and organizations that receive government funding), has also brought accessibility issues to national attention in both the public and private sectors. Accessibility is also being incorporated into official and international standards for usability, such as ANSI 200.

Microsoft® Windows® 95 incorporates many new features and enhancements designed to make the operating system more accessible to people with disabilities. Applications should also follow accessible design practices to make computing in the home, schools, and workplace more accessible to everyone.

This document discusses how computers can be made accessible for people with disabilities and describes how to design and produce computer software that accommodates users with disabilities.

Using This Accessibility Document

This document contains the following sections:

- A summary of accessibility features and programming techniques.
- Background information about disabilities and accessibility aids.
- Detailed descriptions of features and programming techniques that make applications more accessible or compatible with accessibility aids.
- A list of additional resources for information about accessibility aids and their manufacturers.
- Guidelines for developing applications for Windows version 3.*x* so that they are more accessible.

The guidelines discussed in this document are not hard and fast rules. However, it is recommended that you adapt or adopt as many as you can to your own applications, or find additional ways to achieve the same goals.

Product designers often start by asking which of the guidelines are the most important, but as a designer, you should determine this yourself based on the current state of your application. First, determine which of the areas your application is already handling correctly and which remain problems. Then decide which problems you can address in the version of your application currently under development. Finally, when you start on the next version of your application, design in corrections to any remaining problems. In this way, you can demonstrate your desire to do "the right thing" in the near term and make a considerable difference in the long term.

Summary of Recommendations

The following lists provide a convenient summary of recommended accessibility features and programming techniques. For detailed explanations of the individual recommendations, see the later sections of this document.

Basic Principles

You should follow these basic principles when designing an accessible application:

- Provide a flexible, customizable user interface for your application that can accommodate the user's needs and preferences. For example, you should allow the user to choose font sizes, reduce visual complexity, and customize the arrangement of menus.
- Make your application's behavior consistent with other Windows-based applications and with system standards. For example, you should support Control Panel settings for colors and sizes and use standard keyboard behavior.
- Support the user's choice of output methods through the use of sound and visuals and of visual text and graphics. You should combine these output methods redundantly or allow the user to choose his or her preferred output method.
- Support the user's choice of input methods by providing keyboard access to all features and by providing access to common tasks using simple mouse operations.
- Use programming techniques and user-interface elements that are compatible with accessibility aids, such as blind access, screen magnification, and voice input utilities.

Keyboard Access

Providing a good keyboard interface is key to designing an accessible application. You should follow these guidelines when implementing keyboard access in your application:

- Provide keyboard access to all features.
- When possible, model your keyboard interface on a familiar application or control.
- Provide access keys for all menu items and controls.
- Use a logical keyboard navigation order.
- Fully document your keyboard user interface.
- If you normally hide some keyboard user interface elements, display them when the Keyboard Preference flag is set.
- Avoid using the **GetAsynchKeyState** function; use alternatives instead, such as the **GetKeyState** function.

Visual Focus

Many accessibility aids need information about the visual focus location. To ensure that this information is available to accessibility aids, you should follow this guideline:

- If you draw the keyboard or visual focus indicator, move the system caret invisibly to track its location.

Controls and Menus

Many accessibility aids require information about controls and menus. You should follow these guidelines when planning an application for use with those aids:

- Use standard controls whenever possible.
- Define correct text labels for standard and owner-drawn controls, even if they will not be visible.
- Include the original class name in the name of a superclassed standard control.
- If you use custom controls, make them OLE Controls or identify them using tooltip controls or invisible text.
- If you use custom controls, support the WM_GETDLGCODE message to identify your control type and keyboard interface.
- If you use owner-drawn menus, provide a textual alternative.
- Display text using the appropriate read-write edit, read-only edit, status, or static controls.

Drawing Operations

Many accessibility aids need to track Windows drawing operations. You should follow these guidelines to ensure compatibility with those aids:

- Draw text using standard graphics device interface (GDI) functions for text output, such as **ExtTextOut**.
- If you use DirectDraw, Display Control Interface (DCI), WinG, or other means of bypassing GDI drawing functions, provide an option that allows the GDI functions to be used instead.

- Label any bitmapped text using tooltip controls or draw the name invisibly when a screen review utility is present.
- Copy and erase text and graphics using standard GDI functions, such as **BitBlt**, **PatBlt**, and **StretchBlt**, rather than modifying memory directly.

Identification of Windows

The following guidelines should be followed when identifying windows for users and accessibility aids:

- If you display a single image representing several objects, identify the separate areas using tooltip controls or by drawing invisibly.
- Provide user-friendly titles for all windows, even if the title is not visible.
- Give functionally unique windows unique window classes.

Timing and Triggering Events

The following guidelines should be followed for timing and navigation:

- Make all user interface timings adjustable or optional.
- If you have messages that time out, allow the user to turn off the time-out behavior.
- Flash text and objects only at the caret blink rate.
- Avoid triggering actions or messages by mouse pointer location.
- Avoid triggering actions or messages by keyboard focus location.

Color

Color is useful to enhance, emphasize, or reiterate information. To ensure that color works for users, you should follow these guidelines:

- Avoid conveying information by color alone or provide an option to convey the information also using text or graphics.
- When screen elements correspond with standard elements, use the system colors chosen in Control Panel.

- When screen elements do not correspond with elements in Control Panel, allow the user to customize the colors.
- Make sure graphic objects are drawn to contrast with the current background color.
- If you display complex backgrounds behind text or objects, allow the user to select a plain background.
- If you use private colors or complex backgrounds, omit them when the High Contrast Mode flag is set.

Size

The size of text and graphics affects usability as well as accessibility. These guidelines should followed to ensure that type size is not a problem for users:

- Allow the user to select font sizes for displayed information.
- If feasible, provide a zoom feature.
- Draw objects using the sizes (screen metrics) selected in Control Panel.
- If you draw lines, determine the proper width rather than using a fixed value.
- Make sure you test display scaling ratios using the Custom Font Size feature.

Sound

Sound should not be the sole means of conveying information. These guidelines should be followed to ensure that sound is used effectively:

- If you provide information by sound alone, present it visually when the ShowSounds flag is set.
- If you generate sounds, provide a way to turn them off.
- Support system sound events, which play optional sounds when the corresponding actions are carried out.
- Define and use as many application-specific sound events as possible.

Layout

These guidelines should be followed to improve the layout of an application for accessibility:

- Attach text labels unambiguously to controls.
- Attach labels to icons when possible or label icons with tooltip controls.
- Try to label controls clearly without relying heavily on context.
- Use consistent screen layouts following Windows user interface guidelines.
- Avoid designing to a specific font.

Optional Ease-of-Use Features

Although not required, the following features are recommended as additional ways to make your application more usable:

- Provide single-click access to as many features as possible and avoid requiring the user to drag, double-click, or use mouse button 2.
- If possible, allow users to customize key commands, menus, and dialog boxes.
- If possible, allow the user to hide graphical decorations so that distractions are reduced.

Verification of an Application's Accessibility

These guidelines should be followed to ensure that an application is truly accessible:

- Test for compatibility with common accessibility aids.
- Include people with disabilities and software vendors in your beta tests.
- Include people with disabilities in your usability tests.
- Determine where your application does or does not follow accessibility guidelines.

Background Information About Accessibility

The following sections provide background about accessibility, including the different types of accessibility aids available and the basic principles underlying accessibility design.

Reasons for Considering Accessibility

Accessibility should be included as part of your application development process for many reasons:

- The potential market is large. There are approximately 49 million people with disabilities in the U. S.; of these people, an estimated 30 million have disabilities that can be affected by the accessibility of computer software. If the international market is included in the totals, the numbers are much higher.
- When you build accessible applications, you are not just targeting people with disabilities, but also their family and friends, their coworkers, and their employers. Large organizations today employ people with disabilities and expect their mainstream computer applications to accommodate all members of their staff. Failing to meet the needs of a relatively small group of users can affect sales to a large organization.
- Legislation requires employers to make reasonable accommodation for employees with disabilities. The Americans with Disabilities Act affects private businesses with 15 or more employees, and section 508 of the Rehabilitation Act affects the federal government and organizations that receive government funding.
- Accessibility is being incorporated into official and international standards for usability, such as ANSI 200.
- Many people develop permanent or temporary disabilities through injury or disease. Repetitive stress injury caused by typing or using a mouse is increasingly common in the computer industry, and as the population ages, more people are likely to experience periods of disability. By age 55, a person has a 25% chance of spending a significant period of time with a serious disability, and the chances increase to 75% by age 70.
- Accessible design actually improves usability for everyone, not just people with disabilities. It helps make applications compatible with

future interface technologies, such as voice recognition. The same techniques that allow applications to be compatible with accessibility aids also help with other forms of application integration, such as compatibility with automated testing tools and other scripting or macro facilities.

Of course, many people believe that accessibility is the right thing to do to allow each individual to live independently and to make his or her maximum contribution to society.

Categories of Disabilities

Individuals are not disabled—rather some people have difficulties performing certain tasks, such as using a mouse or reading small print. When these limitations are serious enough to impact the person's performance, they are referred to as "disabilities." Disabilities can be divided into the following general categories:

- Visual impairments
- Hearing impairments
- Mobility impairments
- Cognitive and language impairments
- Seizure disorders
- Speech impairments

These categories describe groups of disabilities covering a broad range of people with widely different levels of need.

Visual impairments

Visual impairments range from slightly reduced visual acuity to total blindness. Millions of people have vision that is only slightly impaired. They find it difficult to read small print or black text on a gray background, or they experience eyestrain at the end of long computing sessions. These individuals usually do not consider themselves to have a disability.

People whose vision cannot be corrected to better than about 20/80 are described as having low vision. They probably require text to be larger than normal, and they often require a higher contrast between

foreground text and the background. When people's vision cannot be corrected well enough to rely on visual information alone — that is, about 20/200 or higher — they are generally considered to be blind and require displayed information to be converted into spoken text or Braille.

Other types of visual impairments include reduced field of vision, a condition that limits a person's focus to only a small area at time, and color blindness, a condition that makes it difficult or impossible for a person to distinguish certain color combinations. Color blindness affects nearly 10% of males and 1% of females.

Hearing impairments

Some people cannot hear or distinguish different beeps, or recognize spoken words. They may require a program to prompt them in a different manner—for example, through a screen flash or spoken messages displayed as text. Other people can find themselves in this situation when working in a very noisy environment, working in a quiet environment (such as a library) where sound would disturb other people, or working on a machine with broken or missing speakers.

Mobility impairments

Some users are unable to perform certain manual tasks, such as using a mouse or typing two keys at the same time. Others may have a tendency to hit multiple keys, "bounce" fingers off keys, or be unable to hold a printed book. Many users need keyboards and mouse functions to be adapted to their requirements, or they rely exclusively on a single input device.

Cognitive and language impairments

Cognitive impairments take many forms, including short- and long-term memory loss, perceptual differences, and conditions such as Downs syndrome. Language impairments, such as dyslexia or illiteracy, are also very common. Even people learning the language used by their computer software as a second language can be considered to have a form of language impairment. Proper software design can help increase

the number of people with mild cognitive and language impairments who can use computers.

Seizure disorders

People with some forms of epilepsy may experience minor or severe seizures when they see visual signals flash at certain rates or hear certain types of random or repetitive sounds.

Speech impairments

Although difficulty speaking does not normally affect a person's ability to use a computer today, it can be a problem in using telecommunications and voice menus. Difficulty speaking may affect normal computer usage more if voice recognition becomes a common form of input in the future.

Making Computers Accessible

Accessibility means designing computer software to accommodate a wide range of users, including users with disabilities. Although no software can be accessible to everyone, a few simple steps can greatly increase the number of people who can use an application. Special needs can be addressed in the following ways:

- New features can be built into hardware and operating systems that help make them more accessible to users with and without specialized needs. This solution is preferred because the features are available on all workstations and can be used with all "well-behaved" applications (that is, applications that follow standard programming practices and implement the guidelines described in this document). However, some issues cannot be addressed at a system level.
- Accessibility aids can modify the system making it usable by more people. These aids work with all well-behaved applications, but they are not available when the user moves to a new computer. In addition, many people who could benefit from such aids never obtain them.
- Specialized applications, such as word processors, can be designed to integrate voice and text to help individuals with limited reading and

writing skills. A few of these applications are available, but in general, people with disabilities need to use mainstream applications to cooperate with their coworkers and to take advantage of the latest mainstream features.

- Usability features can be built into mainstream applications, making them easier for people with disabilities to use. Examples include customizable colors and access keys, which can only be provided by the application itself. Often, these features also benefit people who do not have disabilities. Mainstream applications can also be made compatible with accessibility aids.

Types of Accessibility Aids

The following sections describe some of the more common types of accessibility aids—utilities that are added to a computer to make it more accessible to people with certain disabilities. Not all people with disabilities use tools like these, but being familiar with them and how they work can help you to design and build applications that work well with them.

Screen enlargement utilities

A screen enlarger (also called a screen magnifier or large print program) is a utility that allows the user to magnify a portion of his or her screen, effectively turning the computer monitor into a window that shows a portion of an enlarged virtual display. The user can use mouse or keyboard commands to move the window to view different areas of the virtual display. An enlarger needs to track where the user is working, so it can automatically keep the active area in view.

Screen review utilities

People who are blind use the computer with the aid of a screen review utility (also called a blind access utility or screen reader). A screen review utility takes the information displayed visually on the screen and makes it available through alternative, nonvisual media, such as synthesized speech or a refreshable Braille display. Because both of those media present only text, not graphics, the utility needs to render screen elements as text—for example, by assigning a user-friendly

name to each graphic object. The utility also needs to track what the user is doing and the location of the focus to be able to describe the important aspects of what is happening on the screen.

In Windows and other graphical environments, a screen review utility works by watching all operations that draw information to the screen. Then it builds up an "off-screen model," a database of objects on the screen, their properties, and their spatial relationships. Some information will be automatically read to the user when it changes on the screen, and other information will be found when the user requests it. A screen review utility often accepts configuration files (also called set files or profiles), which tell it how to work correctly with particular applications.

Voice input utilities

People with severe mobility impairments will often use a voice input utility (also called a speech recognition program) to control the computer with their voice instead of the mouse and keyboard. This kind of utility is also increasingly being used to boost productivity of people who do not have disabilities.

Like a screen review utility, a voice input utility tries to identify objects on the screen that can be manipulated and to determine appropriate names for them so that the user can activate an object with a single phrase. It also needs to be able to manipulate controls programmatically and be able to detect what the user is doing and the changes that result.

On-screen keyboards

Some people with motion impairments cannot use a standard keyboard, but they may be able to use another input method, such as a switch or a pointing device. An on-screen keyboard is a utility that displays a list of commands to the user and allows him or her to choose and execute the commands using a variety of input methods. A common use of this technique is to display a set of keys that the user can point and click on to type into the computer. Variations of this technique include Morse-code input systems and single- or double-switch input systems.

If a single-switch system is used, an on-screen keyboard successively highlights groups of commands until the user selects one group by

pressing a switch. Then the utility successively highlights smaller groups of commands within the selected group until the user selects the specific command to run. If a user can point but not click, he or she can activate a command using a head pointer by pausing the pointer over the command for a certain amount of time.

An on-screen keyboard is also commonly used to display buttons with all the commands available at a given time. To display them, the utility needs to identify, name, and activate controls much the way a voice input utility does.

Keyboard filters

Impaired dexterity may make it difficult for a person to use a standard keyboard, but keyboard filters built into Windows 95 compensate somewhat by correcting for erratic motion, tremors, slow response time, and similar conditions. In most cases, however, it is not possible to apply the same corrections to pointing devices, such as the mouse, so users with impaired dexterity are restricted to keyboard input. Other types of keyboard filters include typing aids, such as word prediction and abbreviation expansion utilities, and add-on spelling checkers. These typing aids are also increasingly being used to improve the typing speed and accuracy of users who do not have disabilities.

Designing Accessible Applications

The remaining sections of this document describe specific guidelines that can be used when designing a mainstream application for use by individuals with disabilities. The guidelines fall into two categories: end-user features that make the application more usable and programming techniques that make the application compatible with accessibility aids. The following sections expand on the lists provided earlier in this document.

It is impossible for any written guidelines to cover every situation, so it is worth restating the basic principles that underlie accessible design:

- Provide a flexible, customizable user interface for your application that can accommodate the user's needs and preferences. For example,

you should allow the user to choose font sizes, reduce visual complexity, and customize the arrangement of menus.

- Make your application's behavior consistent with other Windows-based applications and with system standards. For example, you should support Control Panel settings for colors and sizes and follow standard keyboard behavior.
- Support the user's choice of output methods through the use of sound and visuals and of visual text and graphics. You should combine these output methods redundantly or allow the user to choose his or her preferred output method.
- Support the user's choice of input methods by providing keyboard access to all features and by providing access to common tasks using simple mouse operations.
- Use programming techniques and user-interface elements that are compatible with accessibility aids, such as blind access, screen magnification, and voice input utilities.

By keeping these principles in mind and by following the specific recommendations in this document, you should be able to address most issues encountered when designing an application for accessibility.

Keyboard Access

An application should be designed so that a mouse is not required for its use. Providing a good keyboard interface is actually one of the most important and most visible aspects of software accessibility. It is important because it affects users with a wide range of disabilities. For example, a keyboard interface may be the only option for users who are blind, who use voice input or keyboard macro utilities, or who cannot use a mouse. (The accessibility options in Windows 95 can often compensate for problems involving the keyboard, but it is more difficult to handle problems relating to pointing devices. Although Windows supports movement of the mouse pointer using the keyboard, this technique cannot be used by everyone, and it is extremely cumbersome at best.)

In addition to accommodating people with disabilities, there are many other reasons for providing a good keyboard interface. Many

experienced typists do not like to take their hands off the keyboard to use a mouse. In addition, pointing devices on many laptop computers can be inconvenient to use, and a working mouse may not always be available.

Choosing a Keyboard Interface

The key to defining a good keyboard interface is to adapt models already familiar to the user. To accomplish that, you should make the interface keystroke-compatible with other familiar applications or controls.

Here are some examples showing how the user can move and resize objects using the keyboard:

- **Menus and Dialog Boxes.** Menus are one of the most common and standardized user interface elements. Putting commands on the menu is always a safe option. However, doing so consistently may make menus large and unwieldy. To prevent these types of problems, you can provide a default configuration that hides most of the menu items for your application's commands and only shows them when the user explicitly requests them through an option in your application or when the user sets the Keyboard Preference flag in Control Panel (discussed later in this document). It is usually possible to provide dialog boxes or property sheets with the same functionality used by the mouse.
- **Property Sheets.** The user can directly manipulate controls to adjust their location and size using the object's property sheet. This method is used by the Microsoft Visual Basic® programming system.

The following examples show some navigational techniques that you may be able to adopt and adapt to your application:

- **Move and Size Commands.** Most windows have Move and Size commands on their system menu, which enable the user to perform those operations using the keyboard. The same commands with the same user interface can also be provided for application-specific objects, either on the normal menu bar or on the object's context menu.

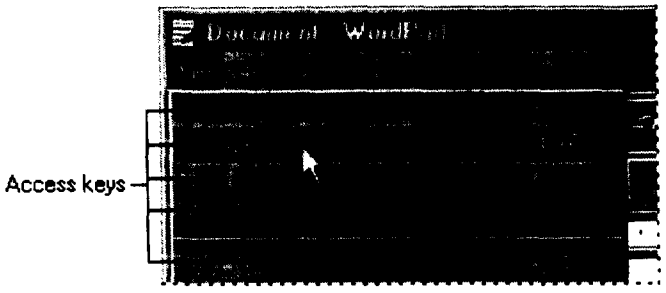
- **Navigating Between Objects.** Windows Help uses a common method for navigating between objects. Windows Help allows the user to press the TAB key to move the focus through a list of "hot spots" or active regions, and to press the ENTER key to choose the currently selected active region. The SHIFT+TAB key combination is used to move backwards through the list, and arrow keys are used to move in specific directions (this capability is especially useful if the items are arranged in two dimensions). An application using this method can also allow the user to type one or more letters to move to the next active region whose label starts with those letters, much like the user can move through entries in a standard list box.
- **Navigating Between Window Areas.** Some applications divide their window into two or more panes, separate areas showing different information that the user can move between using a simple keystroke. For example, Windows Explorer can display a single window with three panes. The TAB key or the SHIFT+TAB key combination moves the focus between the panes (that is, the tree view pane, the folder pane, and the tool bar), and the arrow keys move the focus around within a pane. Multiple Document Interface (MDI) follows another convention, the use of the F6 key and the CTRL+F6 key combination to move the focus between panes and child windows; it also presents a list of windows and allows the user to move between them using the Windows menu.

  Some applications use the CTRL+TAB or CTRL+PAGE DOWN key combination to shift through groups of panes or pages.

Providing Access Keys

Underlined letters on a menu or control, known as access keys (also called shortcut keys or quick-access characters), should exist for all menu items and controls.

The following illustration shows some access keys on a File menu.



Once users become familiar with an application, they become more likely to use access keys to speed up common operations. This tendency is even more common among users who do not use a mouse, including those who are blind. For example, screen review utilities present the user interface sequentially, so a user has to press the TAB key and read or listen to find out what he or she has reached before deciding whether to press the TAB key again. This mechanism slows down the process of locating the correct destination and makes use of access keys much more attractive.

You can omit access keys in these two situations:

- There are not any unused characters in the label to use. In this case, you can either rename the control or omit the access key, depending on how often or how repetitively the command will be used. An example is the standard Sort Options dialog box; it has identical sets of controls for each of the three sort criteria, which makes it difficult to assign unique access keys to each.
- The set of commands is very dynamic and cannot be predicted by the user. For example, the standard OK, Cancel, and Apply buttons used in all property sheets cannot be allowed to conflict with controls on a particular page. Because there is no way to predict what controls might be on a page, access keys are not provided for the standard buttons. Another example is a context menu whose commands might vary from one instance to another. Context menu commands can have access keys when it is clear they will not conflict, but they can be omitted when they might conflict with each other or when the user might expect a letter to be associated with one command and find, unpleasantly, that in a particular instance it triggers another.

Another benefit of providing access keys for dialog box controls is that it ensures that a static text control immediately precedes the object it is

labeling in the tab order; this ordering helps screen review utilities determine the relationship between the control and its object.

Using a Logical Keyboard Navigation Order

A logical keyboard navigation order should be used to ensure that dialog boxes and similar groups of objects can be traversed in a logical order using the keyboard, normally from right to left and top to bottom. If the order does not follow this convention, it can be very confusing to users who navigate using the keyboard. It is especially confusing for people who are blind and rely on screen review utilities. Users who are blind explore a dialog box sequentially, instead of scanning the entire box as sighted users would, so random tab order can make the dialog box nearly unusable.

Documenting the Keyboard Interface

Complete documentation for the keyboard user interface should be included with an application. If an application provides keyboard commands and mechanisms, but lacks appropriate documentation for them, the features become essentially useless. If space is a problem, the keyboard interface can be described in another place, and cross-references to the information can be included in the primary documentation. However, keyboard access should not be categorized as a niche or specialty feature, because it is used and relied on by so many people.

Limitations of Keyboard Access

Usable keyboard access should be provided for all features in an application. However, it may not be feasible sometimes. In some cases, it might be too cumbersome for users to use and too challenging for the application designers to implement, especially if the particular feature being considered will be used only occasionally. There are also rare cases where a dialog box is so complex that unique access keys cannot be assigned. In these cases, common sense should dictate where tradeoffs need to be made.

Users can always fall back on tools that enable them to simulate mouse input using the keyboard or other input mechanism, but these tools should not be considered a substitute for good keyboard interface design. For example, a simple drag and drop operation might require 40 or more keystrokes. Operations using that many keystrokes might make an application accessible, but it would certainly not be considered usable or user-friendly. In any case, a user who is blind might still find it difficult to perform such a visual operation using keystrokes. Application designers can design efficient, comprehensive keyboard interfaces for their applications and should make every effort to do so.

Supporting the Keyboard Preference Flag

An application that normally hides some keyboard user interface elements or omits some keyboard mechanisms altogether should present them when the Keyboard Preference flag is set. This flag, which is set by the user in Control Panel, advises an application that the user relies on the keyboard rather than a pointing device, so additional support should be provided where appropriate. An application can test for this flag by calling the **SystemParametersInfo** function with the SPI_GETKEYBOARDPREF value.

Use of the GetAsynchKeyState Function

The **GetAsynchKeyState** function should not be used to retrieve the state of a key or a mouse button, except when absolutely necessary. This function queries the hardware driver to determine the physical key or button state, but ignores any keys being artificially held down or simulated by accessibility aids. When possible, you should use the **GetKeyState** function, which correctly reflects any simulated input. **GetAsynchKeyState** can, however, be called in certain cases, such as when the user types a key to interrupt a lengthy processing task.

When handling mouse drag operations, you should avoid using **GetAsynchKeyState** to detect when a mouse button has been lifted. Instead, you should use the **SetCapture** function and wait for a button-up message. If you use **GetAsynchKeyState**, the results might differ from those obtained using other Windows functions and messages. This